Amendments to the Specification

Please amend the below mentioned paragraphs in the specification as follows:

In the paragraph starting on p. 6, l. 3

However, content aware load balancing implementations still require significant human management and involvement in order to pre-configure data at each server computer system in a coordinated manner with the content aware load balancing device in order to function properly. Further still, as client demands for access to certain data within a group of server computer systems changes over time (e.g., certain data in one server is requested much more frequently than other data from other servers), such conventional content aware load balancing implementations are not able to adapt well to such changing demands. Information system designers have observed that in many client/server information systems, 90 percent of client requests are directed to only 10 percent of the data served by a group of servers. Since information that is most frequently requested by client computer systems can change over a period of time, conventional client/server implementations do not provide an efficient mechanism to dynamically adapt to these demand changes. Such systems rely largely on human intervention to maintain the configuration of the appropriate portions of frequently accessed data stored within multiple server computer systems. ~~Administrator~~ Administrators of such conventional systems can use performance management tools that perform such tasks as data access log analysis to identity those portions of data that are most sought after.

In the paragraph starting on p. 12, l. 3

In addition, in other embodiments of the invention, multiple adaptive load balancers can operate in conjunction with each other to provide dynamic and adaptive replication and migration processing of data such as files across a plurality of server computer systems to allow the system to automatically adjust to changes in demands or requirements to access certain data, files or services. As an example, using two or more adaptive load balancers that each provide access to the virtual file system for a respective set of server files file systems operating in respective groups of server computer systems associated with each adaptive load balancer, if a first adaptive load balancer detects an increase in demand for certain files within its virtual file system, the first adaptive load balancer can replicate (i.e., copy) those

files to the virtual file system of the second adaptive load balancer so that client requests can be served by both adaptive load balancers. This allows a data, file or service serving system using adaptive load balancers of this invention to automatically adapt to changing network conditions. The results result is the ability to efficiently and automatically respond to changes in demands for access to data, files or services without significant human intervention (e.g., without having to manually replicate and then de-replicate copies of files as access demands increase, and then decrease).

In the paragraph starting on p. 13, l. 31

To provide the aggregation of the server file systems ~~155~~ 175 into a single virtual file system 135, the adaptive transaction processor 131 maintains metadata 132 that includes information for mapping or translating the client data access transactions 115 and associated data access operations 117, such as NFS file access requests, into server data access transactions 145. The metadata 132, as will be explained in more detail, contains virtual data system to server data system parameter mapping information. As an example, in embodiments that provide a virtual file system 135, the metadata 132 provides a mapping of virtual file (and directory) handle parameters, specified within an NFS or CIFS file access operation 117 (contained within a client file access transaction 115) received from a client computer system 120, to corresponding server file system file (or directory) handle parameters of a file or directory within a particular server file system 175. The metadata 132 further allows identification or selection of a particular server computer system 170 to which the adaptive transaction processor 131 forwards a resulting mapped server access operation 147 within a server file access transaction 145.

In the paragraph starting on p. 14, l. 16

Within the adaptive load balancer 130, the adaptive transaction processor 131 operates one or more appropriate data (i.e., file) access protocols such as NFS and/or CIFS to communicate with each of the server computer systems 170 to access to the respective server data systems ~~155~~ 175 (i.e., server file systems in this example embodiment). Likewise, the adaptive transaction processor 131 can receive a server transaction response 148 (e.g., an NFS response) from the server computer system 170 (to which the server data access

transaction was forwarded) and can use the metadata 132 to convert, map or otherwise translate this response 148 back into a client data access response 118 that the adaptive load balancer 130 returns to the original requesting client computer system 120. In this manner, in one embodiment the adaptive transaction processor 131 operates to present a collective set of server file systems 175-1 through 175-N that each have a respective set of files 180-1 through 180-N available as an aggregated set of files without identifying to the client computer systems 120 that the files are stored on different server computer systems 170. The adaptive load balancer 130 uses the metadata 132 associated with the virtual file system to manage access to the aggregated sets of files within the server computer systems on behalf of client computer systems requesting access to files via the virtual file system 135. Further operation of an adaptive load balancer 130 in accordance with embodiments of the invention will be explained below in a series of flow charts.

In the paragraph starting on p. 17, l. 11

In step 206, the adaptive load balancer processes the client data ~~acces~~ access transaction 115 using the metadata 132 to create a server data access transaction 145 that is transferred to a server computer system 170 to carry out the data access operation 117 specified by the client computer system 120. Further details of the processing performed in step 206 will be explained shortly in Figure 3.

In the paragraph starting on p. 17, l. 25:

Figure 3 is a flow chart of processing steps performed in one embodiment of the invention to ~~processes~~ process the client data access transaction 115 using the metadata 132 to create a server data access transaction 145 that is transferred to a server computer system 170 to carry out the data access operation 117 specified by the client computer system 120. In other words, the steps in Figure 3 are sub-steps performed in step 206 of Figure 2.

In the paragraph starting on p. 21, l. 14

EXAMPLE VIRTUAL FILE SYSTEM ~~DIRETORY~~ DIRECTORY STRUCTURE

In the paragraph starting on p. 22, l. 5

As an example, a client ~~is~~ seeking access (e.g., via a sequence of NFS lookup commands) to a file having the full directory path /home/usr/joe/JoeFilel.doc can provide a sequence of one or more NFS LOOKUP operations 117 in the adaptive transaction process 131 and can use the above example directory location mapping table to identify the virtual file handle (VFH) "12345" associated with that file in the virtual file system 135. Note that the example file handles "12345" and so forth represented here as examples only and do not represent actual file handles used by NFS of CIFS.

In the paragraph starting on p. 22, l. 25

EXAMPLE DIRECTORY FILE MAPPING TABLE

The example directory file mapping table above maps virtual file handles of files or directors within the virtual file system 135 to physical or server file handles of actual files stored within server computer systems. In addition, example directory file mapping table includes the identity of one or more server computer systems 170 ~~at~~ that contain an instance or copy of that file corresponding to the virtual file handle in column 1. Note that in this example, the file JoeFile2.doc is known to client applications 125 by using the virtual file handle 34567 and maps to two replicated server instances or copies of that file. The first copy has a physical file handle "CDEFG" and resides within the server file system 175-1 within the server computer system 170-1 in Figure 1 whereas the second copy has a physical file handle "PQRST" and resides within the server file system 175-2 operating within the server computer system 170-2.

In the paragraph starting on p. 25, l. 6

In step 304, the adaptive load balancer receives the client file access transaction 115. As an example, suppose client application 125-2 operating client computer system ~~125-2~~ 120-2 transmits a client file access transaction 115 that specifies a data access operation 117 such as in NFS READ command to read from the file /home/usr/joe/JoeFile2.doc. This

command would appear as "NFS READ 34567", by way of example only. Note that according to standard operation of the NFS protocol, to obtain the virtual file handle 34567 of the file JoeFile2.doc, the client computer system ~~125-2~~ 120-2 would have to had transmitted a series of NFS LOOKUP operations on subsequent portions of the directory path /home/usr/Joe/Joefile2.doc (i.e., performed NFS LOOKUP operations first on /home, then on /usr, then on /Joe, and then finally on JoeFile2.doc) and the adaptive transaction processor 131 would use the metadata 132 explained above to return corresponding file handles in response to each NFS lookup operation. Thus the example of NFS READ 34567 assumes this process has already taken place (and would have been performed as explained below for metadata-only data access operations 117).

In the paragraph starting on p. 26, l. 1

In step 307, if the active transaction table does not contain an active transaction table entry (i.e., row) that corresponds to (i.e., that contains) the client transaction identity CT-2, the adaptive load balancer 130 creates an active transaction table entry (i.e., adds a new row or record to the table) containing the assigned client transaction identity CT-2 along with an identity of the client computer system ~~125-2~~ 120-2 (in column 2 of the above example active transaction table) from which the client file access transaction 115 was received. Also in this step, the transaction table entry can be populated (by the adaptive transaction process 131) with client specific information such as NFS attributes, flags, a client file handle, and so forth to speed up reverse translation of server responses explained later. The adaptive transaction process 131 can store this information in the virtual file parameters column (shown in the example table above) for later use.

In the paragraph starting on p. 27, l. 8

In step 312, the adaptive load balancer 130 matches the virtual file identifier (e.g., 34567) specified in the client file access transaction 115 to a matching forwarding table entry in a forwarding table (e.g., the directory file mapping table that maps virtual file handles to physical or server file handles) to identify a corresponding physical file identifier contained in the matching forwarding table entry. Note that when matching a directory, the directory

location mapping table is used. The NFS protocol can indicate whether or not the file handles are associated with the directory or a file.

In the paragraph starting on p. 29, l. 3

In step 317, the adaptive transaction processor 131 can use various criteria to select which particular copy of the file will be chosen for application of the client's requested data access operation 117 ~~win~~ with more than one directory file mapping table entry matches the virtual file handle. Factors that can be used in making a selection between which physical file handle to use (and thus which choice of server computer system to select) can include such things as performance metrics associated with each server computer system containing a copy of the file, network congestion that might be experienced when communicating with each server computer system 170 containing a copy of file, and other considerations. Assume for the example command NFS READ 34567 that the adaptive transaction processor 131 selects the matching entry in the directory file mapping table corresponding to the file handle PQRST (i.e., the second matching entry) for the server computer system 170-2. The newly create server data access transaction 145 thus appears as "NFS READ PQRST."

In the paragraph starting on p. 30, l. 24

Figure 8 is a flow chart of processing steps that the adaptive load balancer 130 performs in accordance with embodiments of the invention to ~~translated~~ translate server transaction response 148 into a client data access response 116 ~~or~~ for transmission to a client computer system 120. Generally, the processing in Figure 8 performs a reverse translation of the server transaction response 148 using the above explained data structures and tables as explained above to produce a corresponding client data access response 116.

In the paragraph starting on p. 33, l. 20

Alternatively, the replication can be ~~carryout~~ carried out by having the adaptive load balancer 130-1 ~~appeared~~ appear as a client computer system to the adaptive load balancer 130-2 and provide a sequence of access commands to cause creation of the file 181 within an

appropriate server computer system 170-4 through 170-6 under management of the second adaptive load balancer 130-2. In other words, the first scenario the first adaptive load balancer 130-1 copies the file itself to the second server computer system 170-4 and then informs the second adaptive load balancer 130-2 so that the metadata 132-2 is properly updated. In the second or alternative scenario, the first adaptive load balancer 130-1 operate as a client with respect to the second adaptive load balancer 130-2 thus causing the second adaptive load balancer 132 to create a second instance of the file 180-1. In this manner, since multiple copies of the file for which frequent access was detected are now distributed within multiple locations, client computer systems are offered access to this file from multiple adaptive load balancers ~~must~~ thus reducing the load or demand for the file from the first adaptive load balancer 131 the detected frequent access to the file.

In the paragraph starting on p. 38, l. 20

Figure 13 is a flow chart of processing steps that the adaptive load balancer 130 performs in accordance with embodiments of the invention to process transactions that ~~you~~ require access to files within server file systems 175 in situations in which replicated copies of such files exist. As an example, the processing of the flow chart steps in Figure 13 can be used to handle situations in which an NFS WRITE data access operation 117 is received that requires writing data to a file that contains multiple replicated copies within the virtual file system 135.

In the paragraph starting on p. 39, l. 30

In step 560 in this example embodiment, the file access operation specified by the client file access transaction is a file access operation that requires modification of a file within the virtual file system by creation of a server file access transaction 145 for into a specific server computer system containing a selected version of the file (that may or may not be replicated). Note ~~is~~ if the files are replicated, during the processing of server selection using the metadata 132 as explained above, one instance of the file can be designated as a master copy of the file and the server file system 175 that maintains a master copy would be

selected as the file system to which the server data access transaction 145 is transmitted or forwarded.

In the paragraph starting on p. 41, l. 5

In an alternative configuration, in step 568, if replicated copies of the file that was modified exist within different server file systems 175, then the adaptive load balancer 130 deletes each replicated copy of the file within respective server file systems 175 of server computer systems 170 that maintain a replicated copy of that file, such that file that was modified in the first server file system 175-1 is the only copy of that file that remains accessible to client computer systems 120 within the virtual file system 135. In this manner, if a replicated file exists ~~in~~ and a modification is made to one instance of the replicated file, this embodiment of the invention causes the remaining replicated copies of the file to be deleted so that all accesses will now be directed to the modified version and no replicated copies will exist. Such embodiment would primarily be used for read-only file systems in which modification to replicated files would really take place.